

Amazon

Exam Questions AWS-Certified-Data-Engineer-Associate

AWS Certified Data Engineer - Associate (DEA-C01)



NEW QUESTION 1

A data engineer needs to join data from multiple sources to perform a one-time analysis job. The data is stored in Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3.

Which solution will meet this requirement MOST cost-effectively?

- A. Use an Amazon EMR provisioned cluster to read from all source
- B. Use Apache Spark to join the data and perform the analysis.
- C. Copy the data from DynamoDB, Amazon RDS, and Amazon Redshift into Amazon S3. Run Amazon Athena queries directly on the S3 files.
- D. Use Amazon Athena Federated Query to join the data from all data sources.
- E. Use Redshift Spectrum to query data from DynamoDB, Amazon RDS, and Amazon S3 directly from Redshift.

Answer: C

Explanation:

Amazon Athena Federated Query is a feature that allows you to query data from multiple sources using standard SQL. You can use Athena Federated Query to join data from Amazon DynamoDB, Amazon RDS, Amazon Redshift, and Amazon S3, as well as other data sources such as MongoDB, Apache HBase, and Apache Kafka¹. Athena Federated Query is a serverless and interactive service, meaning you do not need to provision or manage any infrastructure, and you only pay for the amount of data scanned by your queries. Athena Federated Query is the most cost-effective solution for performing a one-time analysis job on data from multiple sources, as it eliminates the need to copy or move data, and allows you to query data directly from the source.

The other options are not as cost-effective as Athena Federated Query, as they involve additional steps or costs. Option A requires you to provision and pay for an Amazon EMR cluster, which can be expensive and time-consuming for a one-time job. Option B requires you to copy or move data from DynamoDB, RDS, and Redshift to S3, which can incur additional costs for data transfer and storage, and also introduce latency and complexity. Option D requires you to have an existing Redshift cluster, which can be costly and may not be necessary for a one-time job. Option E also does not support querying data from RDS directly, so you would need to use Redshift Federated Query to access RDS data, which adds another layer of complexity². References:

? Amazon Athena Federated Query

? Redshift Spectrum vs Federated Query

NEW QUESTION 2

A company uses an Amazon QuickSight dashboard to monitor usage of one of the company's applications. The company uses AWS Glue jobs to process data for the dashboard. The company stores the data in a single Amazon S3 bucket. The company adds new data every day.

A data engineer discovers that dashboard queries are becoming slower over time. The data engineer determines that the root cause of the slowing queries is long-running AWS Glue jobs.

Which actions should the data engineer take to improve the performance of the AWS Glue jobs? (Choose two.)

- A. Partition the data that is in the S3 bucket
- B. Organize the data by year, month, and day.
- C. Increase the AWS Glue instance size by scaling up the worker type.
- D. Convert the AWS Glue schema to the DynamicFrame schema class.
- E. Adjust AWS Glue job scheduling frequency so the jobs run half as many times each day.
- F. Modify the 1AM role that grants access to AWS glue to grant access to all S3 features.

Answer: AB

Explanation:

Partitioning the data in the S3 bucket can improve the performance of AWS Glue jobs by reducing the amount of data that needs to be scanned and processed. By organizing the data by year, month, and day, the AWS Glue job can use partition pruning to filter out irrelevant data and only read the data that matches the query criteria. This can speed up the data processing and reduce the cost of running the AWS Glue job. Increasing the AWS Glue instance size by scaling up the worker type can also improve the performance of AWS Glue jobs by providing more memory and CPU resources for the Spark execution engine. This can help the AWS Glue job handle larger data sets and complex transformations more efficiently. The other options are either incorrect or irrelevant, as they do not affect the performance of the AWS Glue jobs. Converting the AWS Glue schema to the DynamicFrame schema class does not improve the performance, but rather provides additional functionality and flexibility for data manipulation. Adjusting the AWS Glue job scheduling frequency does not improve the performance, but rather reduces the frequency of data updates. Modifying the IAM role that grants access to AWS Glue does not improve the performance, but rather affects the security and permissions of the AWS Glue service. References:

? Optimising Glue Scripts for Efficient Data Processing: Part 1 (Section: Partitioning Data in S3)

? Best practices to optimize cost and performance for AWS Glue streaming ETL jobs (Section: Development tools)

? Monitoring with AWS Glue job run insights (Section: Requirements)

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 133)

NEW QUESTION 3

A data engineer is configuring Amazon SageMaker Studio to use AWS Glue interactive sessions to prepare data for machine learning (ML) models.

The data engineer receives an access denied error when the data engineer tries to prepare the data by using SageMaker Studio.

Which change should the engineer make to gain access to SageMaker Studio?

- A. Add the AWSGlueServiceRole managed policy to the data engineer's IAM user.
- B. Add a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy.
- C. Add the AmazonSageMakerFullAccess managed policy to the data engineer's IAM user.
- D. Add a policy to the data engineer's IAM user that allows the sts:AddAssociation action for the AWS Glue and SageMaker service principals in the trust policy.

Answer: B

Explanation:

This solution meets the requirement of gaining access to SageMaker Studio to use AWS Glue interactive sessions. AWS Glue interactive sessions are a way to use AWS Glue DataBrew and AWS Glue Data Catalog from within SageMaker Studio. To use AWS Glue interactive sessions, the data engineer's IAM user needs to have permissions to assume the AWS Glue service role and the SageMaker execution role. By adding a policy to the data engineer's IAM user that includes the sts:AssumeRole action for the AWS Glue and SageMaker service principals in the trust policy, the data engineer can grant these permissions and avoid the access denied error. The other options are not sufficient or necessary to resolve the error. References:

? Get started with data integration from Amazon S3 to Amazon Redshift using AWS Glue interactive sessions

? Troubleshoot Errors - Amazon SageMaker

? AccessDeniedException on sagemaker:CreateDomain in AWS SageMaker Studio, despite having SageMakerFullAccess

NEW QUESTION 4

A data engineer needs to schedule a workflow that runs a set of AWS Glue jobs every day. The data engineer does not require the Glue jobs to run or finish at a specific time.

Which solution will run the Glue jobs in the MOST cost-effective way?

- A. Choose the FLEX execution class in the Glue job properties.
- B. Use the Spot Instance type in Glue job properties.
- C. Choose the STANDARD execution class in the Glue job properties.
- D. Choose the latest version in the GlueVersion field in the Glue job properties.

Answer: A

Explanation:

The FLEX execution class allows you to run AWS Glue jobs on spare compute capacity instead of dedicated hardware. This can reduce the cost of running non-urgent or non-time sensitive data integration workloads, such as testing and one-time data loads. The FLEX execution class is available for AWS Glue 3.0 Spark jobs. The other options are not as cost-effective as FLEX, because they either use dedicated resources (STANDARD) or do not affect the cost at all (Spot Instance type and GlueVersion). References:

? Introducing AWS Glue Flex jobs: Cost savings on ETL workloads

? Serverless Data Integration – AWS Glue Pricing

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 125)

NEW QUESTION 5

A company's data engineer needs to optimize the performance of table SQL queries. The company stores data in an Amazon Redshift cluster. The data engineer cannot increase the size of the cluster because of budget constraints.

The company stores the data in multiple tables and loads the data by using the EVEN distribution style. Some tables are hundreds of gigabytes in size. Other tables are less than 10 MB in size.

Which solution will meet these requirements?

- A. Keep using the EVEN distribution style for all table
- B. Specify primary and foreign keys for all tables.
- C. Use the ALL distribution style for large table
- D. Specify primary and foreign keys for all tables.
- E. Use the ALL distribution style for rarely updated small table
- F. Specify primary and foreign keys for all tables.
- G. Specify a combination of distribution, sort, and partition keys for all tables.

Answer: C

Explanation:

This solution meets the requirements of optimizing the performance of table SQL queries without increasing the size of the cluster. By using the ALL distribution style for rarely updated small tables, you can ensure that the entire table is copied to every node in the cluster, which eliminates the need for data redistribution during joins. This can improve query performance significantly, especially for frequently joined dimension tables. However, using the ALL distribution style also increases the storage space and the load time, so it is only suitable for small tables that are not updated frequently or extensively. By specifying primary and foreign keys for all tables, you can help the query optimizer to generate better query plans and avoid unnecessary scans or joins. You can also use the AUTO distribution style to let Amazon Redshift choose the optimal distribution style based on the table size and the query patterns. References:

? Choose the best distribution style

? Distribution styles

? Working with data distribution styles

NEW QUESTION 6

A data engineer must manage the ingestion of real-time streaming data into AWS. The data engineer wants to perform real-time analytics on the incoming streaming data by using time-based aggregations over a window of up to 30 minutes. The data engineer needs a solution that is highly fault tolerant.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Use an AWS Lambda function that includes both the business and the analytics logic to perform time-based aggregations over a window of up to 30 minutes for the data in Amazon Kinesis Data Streams.
- B. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data that might occasionally contain duplicates by using multiple types of aggregations.
- C. Use an AWS Lambda function that includes both the business and the analytics logic to perform aggregations for a tumbling window of up to 30 minutes, based on the event timestamp.
- D. Use Amazon Managed Service for Apache Flink (previously known as Amazon Kinesis Data Analytics) to analyze the data by using multiple types of aggregations to perform time-based analytics over a window of up to 30 minutes.

Answer: A

Explanation:

This solution meets the requirements of managing the ingestion of real-time streaming data into AWS and performing real-time analytics on the incoming streaming data with the least operational overhead. Amazon Managed Service for Apache Flink is a fully managed service that allows you to run Apache Flink applications without having to manage any infrastructure or clusters. Apache Flink is a framework for stateful stream processing that supports various types of aggregations, such as tumbling, sliding, and session windows, over streaming data. By using Amazon Managed Service for Apache Flink, you can easily connect to Amazon Kinesis Data Streams as the source and sink of your streaming data, and perform time-based analytics over a window of up to 30 minutes. This solution is also highly fault tolerant, as Amazon Managed Service for Apache Flink automatically scales, monitors, and restarts your Flink applications in case of failures. References:

? Amazon Managed Service for Apache Flink

? Apache Flink

? Window Aggregations in Flink

NEW QUESTION 7

A company created an extract, transform, and load (ETL) data pipeline in AWS Glue. A data engineer must crawl a table that is in Microsoft SQL Server. The data

engineer needs to extract, transform, and load the output of the crawl to an Amazon S3 bucket. The data engineer also must orchestrate the data pipeline. Which AWS service or feature will meet these requirements MOST cost-effectively?

- A. AWS Step Functions
- B. AWS Glue workflows
- C. AWS Glue Studio
- D. Amazon Managed Workflows for Apache Airflow (Amazon MWAA)

Answer: B

Explanation:

AWS Glue workflows are a cost-effective way to orchestrate complex ETL jobs that involve multiple crawlers, jobs, and triggers. AWS Glue workflows allow you to visually monitor the progress and dependencies of your ETL tasks, and automatically handle errors and retries. AWS Glue workflows also integrate with other AWS services, such as Amazon S3, Amazon Redshift, and AWS Lambda, among others, enabling you to leverage these services for your data processing workflows. AWS Glue workflows are serverless, meaning you only pay for the resources you use, and you don't have to manage any infrastructure.

AWS Step Functions, AWS Glue Studio, and Amazon MWAA are also possible options for orchestrating ETL pipelines, but they have some drawbacks compared to AWS Glue workflows. AWS Step Functions is a serverless function orchestrator that can handle different types of data processing, such as real-time, batch, and stream processing. However, AWS Step Functions requires you to write code to define your state machines, which can be complex and error-prone. AWS Step Functions also charges you for every state transition, which can add up quickly for large-scale ETL pipelines.

AWS Glue Studio is a graphical interface that allows you to create and run AWS Glue ETL jobs without writing code. AWS Glue Studio simplifies the process of building, debugging, and monitoring your ETL jobs, and provides a range of pre-built transformations and connectors. However, AWS Glue Studio does not support workflows, meaning you cannot orchestrate multiple ETL jobs or crawlers with dependencies and triggers. AWS Glue Studio also does not support streaming data sources or targets, which limits its use cases for real-time data processing.

Amazon MWAA is a fully managed service that makes it easy to run open-source versions of Apache Airflow on AWS and build workflows to run your ETL jobs and data pipelines. Amazon MWAA provides a familiar and flexible environment for data engineers who are familiar with Apache Airflow, and integrates with a range of AWS services such as Amazon EMR, AWS Glue, and AWS Step Functions. However, Amazon MWAA is not serverless, meaning you have to provision and pay for the resources you need, regardless of your usage. Amazon MWAA also requires you to write code to define your DAGs, which can be challenging and time-consuming for complex ETL pipelines. References:

- ? AWS Glue Workflows
- ? AWS Step Functions
- ? AWS Glue Studio
- ? Amazon MWAA
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 8

A company is planning to upgrade its Amazon Elastic Block Store (Amazon EBS) General Purpose SSD storage from gp2 to gp3. The company wants to prevent any interruptions in its Amazon EC2 instances that will cause data loss during the migration to the upgraded storage.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Create snapshots of the gp2 volume
- B. Create new gp3 volumes from the snapshot
- C. Attach the new gp3 volumes to the EC2 instances.
- D. Create new gp3 volume
- E. Gradually transfer the data to the new gp3 volume
- F. When the transfer is complete, mount the new gp3 volumes to the EC2 instances to replace the gp2 volumes.
- G. Change the volume type of the existing gp2 volumes to gp3. Enter new values for volume size, IOPS, and throughput.
- H. Use AWS DataSync to create new gp3 volume
- I. Transfer the data from the original gp2 volumes to the new gp3 volumes.

Answer: C

Explanation:

Changing the volume type of the existing gp2 volumes to gp3 is the easiest and fastest way to migrate to the new storage type without any downtime or data loss. You can use the AWS Management Console, the AWS CLI, or the Amazon EC2 API to modify the volume type, size, IOPS, and throughput of your gp2 volumes. The modification takes effect immediately, and you can monitor the progress of the modification using CloudWatch. The other options are either more complex or require additional steps, such as creating snapshots, transferring data, or attaching new volumes, which can increase the operational overhead and the risk of errors. References:

- ? Migrating Amazon EBS volumes from gp2 to gp3 and save up to 20% on costs (Section: How to migrate from gp2 to gp3)
- ? Switching from gp2 Volumes to gp3 Volumes to Lower AWS EBS Costs (Section: How to Switch from GP2 Volumes to GP3 Volumes)
- ? Modifying the volume type, IOPS, or size of an EBS volume - Amazon Elastic Compute Cloud (Section: Modifying the volume type)

NEW QUESTION 9

A data engineer is building a data pipeline on AWS by using AWS Glue extract, transform, and load (ETL) jobs. The data engineer needs to process data from Amazon RDS and MongoDB, perform transformations, and load the transformed data into Amazon Redshift for analytics. The data updates must occur every hour.

Which combination of tasks will meet these requirements with the LEAST operational overhead? (Choose two.)

- A. Configure AWS Glue triggers to run the ETL jobs even/ hour.
- B. Use AWS Glue DataBrew to clean and prepare the data for analytics.
- C. Use AWS Lambda functions to schedule and run the ETL jobs even/ hour.
- D. Use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift.
- E. Use the Redshift Data API to load transformed data into Amazon Redshift.

Answer: AD

Explanation:

The correct answer is to configure AWS Glue triggers to run the ETL jobs every hour and use AWS Glue connections to establish connectivity between the data sources and Amazon Redshift. AWS Glue triggers are a way to schedule and orchestrate ETL jobs with the least operational overhead. AWS Glue connections are a way to securely connect to data sources and targets using JDBC or MongoDB drivers. AWS Glue DataBrew is a visual data preparation tool that does not support MongoDB as a data source. AWS Lambda functions are a serverless option to schedule and run ETL jobs, but they have a limit of 15 minutes for execution time, which may not be enough for complex transformations. The Redshift Data API is a way to run SQL commands on Amazon Redshift clusters

without needing a persistent connection, but it does not support loading data from AWS Glue ETL jobs. References:

- ? AWS Glue triggers
- ? AWS Glue connections
- ? AWS Glue DataBrew
- ? [AWS Lambda functions]
- ? [Redshift Data API]

NEW QUESTION 10

A company needs to partition the Amazon S3 storage that the company uses for a data lake. The partitioning will use a path of the S3 object keys in the following format: s3://bucket/prefix/year=2023/month=01/day=01.

A data engineer must ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket.

Which solution will meet these requirements with the LEAST latency?

- A. Schedule an AWS Glue crawler to run every morning.
- B. Manually run the AWS Glue CreatePartition API twice each day.
- C. Use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call.
- D. Run the MSCK REPAIR TABLE command from the AWS Glue console.

Answer: C

Explanation:

The best solution to ensure that the AWS Glue Data Catalog synchronizes with the S3 storage when the company adds new partitions to the bucket with the least latency is to use code that writes data to Amazon S3 to invoke the Boto3 AWS Glue create partition API call. This way, the Data Catalog is updated as soon as new data is written to S3, and the partition information is immediately available for querying by other services. The Boto3 AWS Glue create partition API call allows you to create a new partition in the Data Catalog by specifying the table name, the database name, and the partition values¹. You can use this API call in your code that writes data to S3, such as a Python script or an AWS Glue ETL job, to create a partition for each new S3 object key that matches the partitioning scheme.

Option A is not the best solution, as scheduling an AWS Glue crawler to run every morning would introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. AWS Glue crawlers are processes that connect to a data store, progress through a prioritized list of classifiers to determine the schema for your data, and then create metadata tables in the Data Catalog². Crawlers can be scheduled to run periodically, such as daily or hourly, but they cannot run continuously or in real-time. Therefore, using a crawler to synchronize the Data Catalog with the S3 storage would not meet the requirement of the least latency.

Option B is not the best solution, as manually running the AWS Glue CreatePartition API twice each day would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. Moreover, manually running the API would require more operational overhead and human intervention than using code that writes data to S3 to invoke the API automatically.

Option D is not the best solution, as running the MSCK REPAIR TABLE command from the AWS Glue console would also introduce a significant latency between the time new data is written to S3 and the time the Data Catalog is updated. The MSCK REPAIR TABLE command is a SQL command that you can run in the AWS Glue console to add partitions to the Data Catalog based on the S3 object keys that match the partitioning scheme³. However, this command is not meant to be run frequently or in real-time, as it can take a long time to scan the entire S3 bucket and add the partitions. Therefore, using this command to synchronize the Data Catalog with the S3 storage would not meet the requirement of the least latency. References:

- ? AWS Glue CreatePartition API
- ? Populating the AWS Glue Data Catalog
- ? MSCK REPAIR TABLE Command
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 10

A financial services company stores financial data in Amazon Redshift. A data engineer wants to run real-time queries on the financial data to support a web-based trading application. The data engineer wants to run the queries from within the trading application.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Establish WebSocket connections to Amazon Redshift.
- B. Use the Amazon Redshift Data API.
- C. Set up Java Database Connectivity (JDBC) connections to Amazon Redshift.
- D. Store frequently accessed data in Amazon S3. Use Amazon S3 Select to run the queries.

Answer: B

Explanation:

The Amazon Redshift Data API is a built-in feature that allows you to run SQL queries on Amazon Redshift data with web services-based applications, such as AWS Lambda, Amazon SageMaker notebooks, and AWS Cloud9. The Data API does not require a persistent connection to your database, and it provides a secure HTTP endpoint and integration with AWS SDKs. You can use the endpoint to run SQL statements without managing connections. The Data API also supports both Amazon Redshift provisioned clusters and Redshift Serverless workgroups. The Data API is the best solution for running real-time queries on the financial data from within the trading application, as it has the least operational overhead compared to the other options.

Option A is not the best solution, as establishing WebSocket connections to Amazon Redshift would require more configuration and maintenance than using the Data API. WebSocket connections are also not supported by Amazon Redshift clusters or serverless workgroups.

Option C is not the best solution, as setting up JDBC connections to Amazon Redshift would also require more configuration and maintenance than using the Data API. JDBC connections are also not supported by Redshift Serverless workgroups.

Option D is not the best solution, as storing frequently accessed data in Amazon S3 and using Amazon S3 Select to run the queries would introduce additional latency and complexity than using the Data API. Amazon S3 Select is also not optimized for real-time queries, as it scans the entire object before returning the results. References:

- ? Using the Amazon Redshift Data API
- ? Calling the Data API
- ? Amazon Redshift Data API Reference
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 11

A company is planning to migrate on-premises Apache Hadoop clusters to Amazon EMR. The company also needs to migrate a data catalog into a persistent storage solution.

The company currently stores the data catalog in an on-premises Apache Hive metastore on the Hadoop clusters. The company requires a serverless solution to migrate the data catalog.

Which solution will meet these requirements MOST cost-effectively?

- A. Use AWS Database Migration Service (AWS DMS) to migrate the Hive metastore into Amazon S3. Configure AWS Glue Data Catalog to scan Amazon S3 to produce the data catalog.
- B. Configure a Hive metastore in Amazon EM
- C. Migrate the existing on-premises Hive metastore into Amazon EM
- D. Use AWS Glue Data Catalog to store the company's data catalog as an external data catalog.
- E. Configure an external Hive metastore in Amazon EM
- F. Migrate the existing on-premises Hive metastore into Amazon EM
- G. Use Amazon Aurora MySQL to store the company's data catalog.
- H. Configure a new Hive metastore in Amazon EM
- I. Migrate the existing on-premises Hive metastore into Amazon EM
- J. Use the new metastore as the company's data catalog.

Answer: A

Explanation:

AWS Database Migration Service (AWS DMS) is a service that helps you migrate databases to AWS quickly and securely. You can use AWS DMS to migrate the Hive metastore from the on-premises Hadoop clusters into Amazon S3, which is a highlyscalable, durable, and cost-effective object storage service. AWS Glue Data Catalog is a serverless, managed service that acts as a central metadata repository for your data assets. You can use AWS Glue Data Catalog to scan the Amazon S3 bucket that contains the migrated Hive metastore and create a data catalog that is compatible with Apache Hive and other AWS services. This solution meets the requirements of migrating the data catalog into a persistent storage solution and using a serverless solution. This solution is also the most cost-effective, as it does not incur any additional charges for running Amazon EMR or Amazon Aurora MySQL clusters. The other options are either not feasible or not optimal. Configuring a Hive metastore in Amazon EMR (option B) or an external Hive metastore in Amazon EMR (option C) would require running and maintaining Amazon EMR clusters, which would incur additional costs and complexity. Using Amazon Aurora MySQL to store the company's data catalog (option C) would also incur additional costs and complexity, as well as introduce compatibility issues with Apache Hive. Configuring a new Hive metastore in Amazon EMR (option D) would not migrate the existing data catalog, but create a new one, which would result in data loss and inconsistency. References:

? Using AWS Database Migration Service

? Populating the AWS Glue Data Catalog

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 4: Data Analysis and Visualization, Section 4.2: AWS Glue Data Catalog

NEW QUESTION 16

A company stores data from an application in an Amazon DynamoDB table that operates in provisioned capacity mode. The workloads of the application have predictable throughput load on a regular schedule. Every Monday, there is an immediate increase in activity early in the morning. The application has very low usage during weekends.

The company must ensure that the application performs consistently during peak usage times

Which solution will meet these requirements in the MOST cost-effective way?

- A. Increase the provisioned capacity to the maximum capacity that is currently present during peak load times.
- B. Divide the table into two table
- C. Provision each table with half of the provisioned capacity of the original tabl
- D. Spread queries evenly across both tables.
- E. Use AWS Application Auto Scaling to schedule higher provisioned capacity for peak usage time
- F. Schedule lower capacity during off-peak times.
- G. Change the capacity mode from provisioned to on-deman
- H. Configure the table to scale up and scale down based on the load on the table.

Answer: C

Explanation:

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB offers two capacity modes for throughput capacity: provisioned and on-demand. In provisioned capacity mode, you specify the number of read and write capacity units per second that you expect your application to require. DynamoDB reserves the resources to meet your throughput needs with consistent performance. In on-demand capacity mode, you pay per request and DynamoDB scales the resources up and down automatically based on the actual workload. On-demand capacity mode is suitable for unpredictable workloads that can vary significantly over time¹.

The solution that meets the requirements in the most cost-effective way is to use AWS Application Auto Scaling to schedule higher provisioned capacity for peak usage times and lower capacity during off-peak times. This solution has the following advantages:

? It allows you to optimize the cost and performance of your DynamoDB table by adjusting the provisioned capacity according to your predictable workload patterns. You can use scheduled scaling to specify the date and time for the scaling actions, and the new minimum and maximum capacity limits. For example, you can schedule higher capacity for every Monday morning and lower capacity for weekends².

? It enables you to take advantage of the lower cost per unit of provisioned capacity mode compared to on-demand capacity mode. Provisioned capacity mode charges a flat hourly rate for the capacity you reserve, regardless of how much you use. On-demand capacity mode charges for each read and write request you consume, with no minimum capacity required. For predictable workloads, provisioned capacity mode can be more cost-effective than on-demand capacity mode¹.

? It ensures that your application performs consistently during peak usage times by having enough capacity to handle the increased load. You can also use auto scaling to automatically adjust the provisioned capacity based on the actual utilization of your table, and set a target utilization percentage for your table or global secondary index. This way, you can avoid under-provisioning or over- provisioning your table².

Option A is incorrect because it suggests increasing the provisioned capacity to the maximum capacity that is currently present during peak load times. This solution has the following disadvantages:

? It wastes money by paying for unused capacity during off-peak times. If you provision the same high capacity for all times, regardless of the actual workload, you are over-provisioning your table and paying for resources that you don't need¹.

? It does not account for possible changes in the workload patterns over time. If your peak load times increase or decrease in the future, you may need to manually adjust the provisioned capacity to match the new demand. This adds operational overhead and complexity to your application².

Option B is incorrect because it suggests dividing the table into two tables and provisioning each table with half of the provisioned capacity of the original table. This solution has the following disadvantages:

? It complicates the data model and the application logic by splitting the data into two separate tables. You need to ensure that the queries are evenly distributed across both tables, and that the data is consistent and synchronized between them. This adds extra development and maintenance effort to your application³.

? It does not solve the problem of adjusting the provisioned capacity according to the workload patterns. You still need to manually or automatically scale the capacity of each table based on the actual utilization and demand. This may result in under- provisioning or over-provisioning your tables².

Option D is incorrect because it suggests changing the capacity mode from provisioned to on-demand. This solution has the following disadvantages:

? It may incur higher costs than provisioned capacity mode for predictable workloads. On-demand capacity mode charges for each read and write request you consume, with no minimum capacity required. For predictable workloads, provisioned capacity mode can be more cost-effective than on-demand capacity mode,

as you can reserve the capacity you need at a lower rate¹.

? It may not provide consistent performance during peak usage times, as on-demand capacity mode may take some time to scale up the resources to meet the sudden increase in demand. On-demand capacity mode uses adaptive capacity to handle bursts of traffic, but it may not be able to handle very large spikes or sustained high throughput. In such cases, you may experience throttling or increased latency.

References:

? 1: Choosing the right DynamoDB capacity mode - Amazon DynamoDB

? 2: Managing throughput capacity automatically with DynamoDB auto scaling - Amazon DynamoDB

? 3: Best practices for designing and using partition keys effectively - Amazon DynamoDB

? [4]: On-demand mode guidelines - Amazon DynamoDB

? [5]: How to optimize Amazon DynamoDB costs - AWS Database Blog

? [6]: DynamoDB adaptive capacity: How it works and how it helps - AWS Database Blog

? [7]: Amazon DynamoDB pricing - Amazon Web Services (AWS)

NEW QUESTION 21

A company extracts approximately 1 TB of data every day from data sources such as SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. Some of the data sources have undefined data schemas or data schemas that change.

A data engineer must implement a solution that can detect the schema for these data sources. The solution must extract, transform, and load the data to an Amazon S3 bucket. The company has a service level agreement (SLA) to load the data into the S3 bucket within 15 minutes of data creation.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Use Amazon EMR to detect the schema and to extract, transform, and load the data into the S3 bucket
- B. Create a pipeline in Apache Spark.
- C. Use AWS Glue to detect the schema and to extract, transform, and load the data into the S3 bucket
- D. Create a pipeline in Apache Spark.
- E. Create a PySpark program in AWS Lambda to extract, transform, and load the data into the S3 bucket.
- F. Create a stored procedure in Amazon Redshift to detect the schema and to extract, transform, and load the data into a Redshift Spectrum table
- G. Access the table from Amazon S3.

Answer: B

Explanation:

AWS Glue is a fully managed service that provides a serverless data integration platform. It can automatically discover and categorize data from various sources, including SAP HANA, Microsoft SQL Server, MongoDB, Apache Kafka, and Amazon DynamoDB. It can also infer the schema of the data and store it in the AWS Glue Data Catalog, which is a central metadata repository. AWS Glue can then use the schema information to generate and run Apache Spark code to extract, transform, and load the data into an Amazon S3 bucket. AWS Glue can also monitor and optimize the performance and cost of the data pipeline, and handle any schema changes that may occur in the source data. AWS Glue can meet the SLA of loading the data into the S3 bucket within 15 minutes of data creation, as it can trigger the data pipeline based on events, schedules, or on-demand. AWS Glue has the least operational overhead among the options, as it does not require provisioning, configuring, or managing any servers or clusters. It also handles scaling, patching, and security automatically. References:

? AWS Glue

? [AWS Glue Data Catalog]

? [AWS Glue Developer Guide]

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 25

A company receives a daily file that contains customer data in .xls format. The company stores the file in Amazon S3. The daily file is approximately 2 GB in size.

A data engineer concatenates the column in the file that contains customer first names and the column that contains customer last names. The data engineer needs to determine the number of distinct customers in the file.

Which solution will meet this requirement with the LEAST operational effort?

- A. Create and run an Apache Spark job in an AWS Glue notebook
- B. Configure the job to read the S3 file and calculate the number of distinct customers.
- C. Create an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file
- D. Run SQL queries from Amazon Athena to calculate the number of distinct customers.
- E. Create and run an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers.
- F. Use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers.

Answer: D

Explanation:

AWS Glue DataBrew is a visual data preparation tool that allows you to clean, normalize, and transform data without writing code. You can use DataBrew to create recipes that define the steps to apply to your data, such as filtering, renaming, splitting, or aggregating columns. You can also use DataBrew to run jobs that execute the recipes on your data sources, such as Amazon S3, Amazon Redshift, or Amazon Aurora. DataBrew integrates with AWS Glue Data Catalog, which is a centralized metadata repository for your data assets¹.

The solution that meets the requirement with the least operational effort is to use AWS Glue DataBrew to create a recipe that uses the COUNT_DISTINCT aggregate function to calculate the number of distinct customers. This solution has the following advantages:

? It does not require you to write any code, as DataBrew provides a graphical user

interface that lets you explore, transform, and visualize your data. You can use DataBrew to concatenate the columns that contain customer first names and last names, and then use the COUNT_DISTINCT aggregate function to count the number of unique values in the resulting column².

? It does not require you to provision, manage, or scale any servers, clusters, or notebooks, as DataBrew is a fully managed service that handles all the infrastructure for you. DataBrew can automatically scale up or down the compute resources based on the size and complexity of your data and recipes¹.

? It does not require you to create or update any AWS Glue Data Catalog entries, as

DataBrew can automatically create and register the data sources and targets in the Data Catalog. DataBrew can also use the existing Data Catalog entries to access the data in S3 or other sources³.

Option A is incorrect because it suggests creating and running an Apache Spark job in an AWS Glue notebook. This solution has the following disadvantages:

? It requires you to write code, as AWS Glue notebooks are interactive development environments that allow you to write, test, and debug Apache Spark code using Python or Scala. You need to use the Spark SQL or the Spark DataFrame API to read the S3 file and calculate the number of distinct customers.

? It requires you to provision and manage a development endpoint, which is a serverless Apache Spark environment that you can connect to your notebook. You need to specify the type and number of workers for your development endpoint, and monitor its status and metrics.

? It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in your code.

Option B is incorrect because it suggests creating an AWS Glue crawler to create an AWS Glue Data Catalog of the S3 file, and running SQL queries from

Amazon Athena to calculate the number of distinct customers. This solution has the following disadvantages:

? It requires you to create and run a crawler, which is a program that connects to your data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in the Data Catalog. You need to specify the data store, the IAM role, the schedule, and the output database for your crawler.

? It requires you to write SQL queries, as Amazon Athena is a serverless interactive query service that allows you to analyze data in S3 using standard SQL. You need to use Athena to concatenate the columns that contain customer first names and last names, and then use the COUNT(DISTINCT) aggregate function to count the number of unique values in the resulting column.

Option C is incorrect because it suggests creating and running an Apache Spark job in Amazon EMR Serverless to calculate the number of distinct customers.

This solution has the following disadvantages:

? It requires you to write code, as Amazon EMR Serverless is a service that allows you to run Apache Spark jobs on AWS without provisioning or managing any infrastructure. You need to use the Spark SQL or the Spark DataFrame API to read the S3 file and calculate the number of distinct customers.

? It requires you to create and manage an Amazon EMR Serverless cluster, which is a fully managed and scalable Spark environment that runs on AWS Fargate. You need to specify the cluster name, the IAM role, the VPC, and the subnet for your cluster, and monitor its status and metrics.

? It requires you to create or update the AWS Glue Data Catalog entries for the S3 file, either manually or using a crawler. You need to use the Data Catalog as a metadata store for your Spark job, and specify the database and table names in your code.

References:

? 1: AWS Glue DataBrew - Features

? 2: Working with recipes - AWS Glue DataBrew

? 3: Working with data sources and data targets - AWS Glue DataBrew

? [4]: AWS Glue notebooks - AWS Glue

? [5]: Development endpoints - AWS Glue

? [6]: Populating the AWS Glue Data Catalog - AWS Glue

? [7]: Crawlers - AWS Glue

? [8]: Amazon Athena - Features

? [9]: Amazon EMR Serverless - Features

? [10]: Creating an Amazon EMR Serverless cluster - Amazon EMR

? [11]: Using the AWS Glue Data Catalog with Amazon EMR Serverless - Amazon EMR

NEW QUESTION 28

A media company wants to improve a system that recommends media content to customer based on user behavior and preferences. To improve the recommendation system, the company needs to incorporate insights from third-party datasets into the company's existing analytics platform.

The company wants to minimize the effort and time required to incorporate third-party datasets.

Which solution will meet these requirements with the LEAST operational overhead?

A. Use API calls to access and integrate third-party datasets from AWS Data Exchange.

B. Use API calls to access and integrate third-party datasets from AWS

C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories.

D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR).

Answer: A

Explanation:

AWS Data Exchange is a service that makes it easy to find, subscribe to, and use third-party data in the cloud. It provides a secure and reliable way to access and integrate data from various sources, such as data providers, public datasets, or AWS services. Using AWS Data Exchange, you can browse and subscribe to data products that suit your needs, and then use API calls or the AWS Management Console to export the data to Amazon S3, where you can use it with your existing analytics platform. This solution minimizes the effort and time required to incorporate third-party datasets, as you do not need to set up and manage data pipelines, storage, or access controls. You also benefit from the data quality and freshness provided by the data providers, who can update their data products as frequently as needed¹².

The other options are not optimal for the following reasons:

? B. Use API calls to access and integrate third-party datasets from AWS. This option is vague and does not specify which AWS service or feature is used to access and integrate third-party datasets. AWS offers a variety of services and features that can help with data ingestion, processing, and analysis, but not all of them are suitable for the given scenario. For example, AWS Glue is a serverless data integration service that can help you discover, prepare, and combine data from various sources, but it requires you to create and run data extraction, transformation, and loading (ETL) jobs, which can add operational overhead³.

? C. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from AWS CodeCommit repositories. This option is not feasible, as AWS CodeCommit is a source control service that hosts secure Git-based repositories, not a data source that can be accessed by Amazon Kinesis Data Streams.

Amazon Kinesis Data Streams is a service that enables you to capture, process, and analyze data streams in real time, such as clickstream data, application logs, or IoT telemetry. It does not support accessing and integrating data from AWS CodeCommit repositories, which are meant for storing and managing code, not data .

? D. Use Amazon Kinesis Data Streams to access and integrate third-party datasets from Amazon Elastic Container Registry (Amazon ECR). This option is also not feasible, as Amazon ECR is a fully managed container registry service that stores, manages, and deploys container images, not a data source that can be accessed by Amazon Kinesis Data Streams. Amazon Kinesis Data Streams does not support accessing and integrating data from Amazon ECR, which is meant for storing and managing container images, not data .

References:

? 1: AWS Data Exchange User Guide

? 2: AWS Data Exchange FAQs

? 3: AWS Glue Developer Guide

? : AWS CodeCommit User Guide

? : Amazon Kinesis Data Streams Developer Guide

? : Amazon Elastic Container Registry User Guide

? : Build a Continuous Delivery Pipeline for Your Container Images with Amazon ECR as Source

NEW QUESTION 31

A data engineering team is using an Amazon Redshift data warehouse for operational reporting. The team wants to prevent performance issues that might result from long- running queries. A data engineer must choose a system table in Amazon Redshift to record anomalies when a query optimizer identifies conditions that might indicate performance issues.

Which table views should the data engineer use to meet this requirement?

A. STL_USAGE_CONTROL

B. STL_ALERT_EVENT_LOG

C. STL_QUERY_METRICS

D. STL_PLAN_INFO

Answer: B

Explanation:

The STL ALERT EVENT LOG table view records anomalies when the query optimizer identifies conditions that might indicate performance issues. These conditions include skewed data distribution, missing statistics, nested loop joins, and broadcasted data. The STL ALERT EVENT LOG table view can help the data engineer to identify and troubleshoot the root causes of performance issues and optimize the query execution plan. The other table views are not relevant for this requirement. STL USAGE CONTROL records the usage limits and quotas for Amazon Redshift resources. STL QUERY METRICS records the execution time and resource consumption of queries. STL PLAN INFO records the query execution plan and the steps involved in each query. References:

? STL ALERT EVENT LOG

? System Tables and Views

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 32

A data engineer must build an extract, transform, and load (ETL) pipeline to process and load data from 10 source systems into 10 tables that are in an Amazon Redshift database. All the source systems generate .csv, JSON, or Apache Parquet files every 15 minutes. The source systems all deliver files into one Amazon S3 bucket. The file sizes range from 10 MB to 20 GB. The ETL pipeline must function correctly despite changes to the data schema.

Which data pipeline solutions will meet these requirements? (Choose two.)

- A. Use an Amazon EventBridge rule to run an AWS Glue job every 15 minute
- B. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
- C. Use an Amazon EventBridge rule to invoke an AWS Glue workflow job every 15 minute
- D. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successful
- E. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
- F. Configure an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket
- G. Configure an AWS Glue job to process and load the data into the Amazon Redshift table
- H. Create a second Lambda function to run the AWS Glue job
- I. Create an Amazon EventBridge rule to invoke the second Lambda function when the AWS Glue crawler finishes running successfully.
- J. Configure an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket
- K. Configure the AWS Glue workflow to have an on-demand trigger that runs an AWS Glue crawler and then runs an AWS Glue job when the crawler finishes running successful
- L. Configure the AWS Glue job to process and load the data into the Amazon Redshift tables.
- M. Configure an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket
- N. Configure the AWS Glue job to read the files from the S3 bucket into an Apache Spark DataFrame
- O. Configure the AWS Glue job to also put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream
- P. Configure the delivery stream to load data into the Amazon Redshift tables.

Answer: AB

Explanation:

Using an Amazon EventBridge rule to run an AWS Glue job or invoke an AWS Glue workflow job every 15 minutes are two possible solutions that will meet the requirements. AWS Glue is a serverless ETL service that can process and load data from various sources to various targets, including Amazon Redshift. AWS Glue can handle different data formats, such as CSV, JSON, and Parquet, and also support schema evolution, meaning it can adapt to changes in the data schema over time. AWS Glue can also leverage Apache Spark to perform distributed processing and transformation of large datasets. AWS Glue integrates with Amazon EventBridge, which is a serverless event bus service that can trigger actions based on rules and schedules. By using an Amazon EventBridge rule, you can invoke an AWS Glue job or workflow every 15 minutes, and configure the job or workflow to run an AWS Glue crawler and then load the data into the Amazon Redshift tables. This way, you can build a cost-effective and scalable ETL pipeline that can handle data from 10 source systems and function correctly despite changes to the data schema.

The other options are not solutions that will meet the requirements. Option C, configuring an AWS Lambda function to invoke an AWS Glue crawler when a file is loaded into the S3 bucket, and creating a second Lambda function to run the AWS Glue job, is not a feasible solution, as it would require a lot of Lambda invocations and coordination. AWS Lambda has some limits on the execution time, memory, and concurrency, which can affect the performance and reliability of the ETL pipeline. Option D, configuring an AWS Lambda function to invoke an AWS Glue workflow when a file is loaded into the S3 bucket, is not a necessary solution, as you can use an Amazon EventBridge rule to invoke the AWS Glue workflow directly, without the need for a Lambda function. Option E, configuring an AWS Lambda function to invoke an AWS Glue job when a file is loaded into the S3 bucket, and configuring the AWS Glue job to put smaller partitions of the DataFrame into an Amazon Kinesis Data Firehose delivery stream, is not a cost-effective solution, as it would incur additional costs for Lambda invocations and data delivery. Moreover, using Amazon Kinesis Data Firehose to load data into Amazon Redshift is not suitable for frequent and small batches of data, as it can cause performance issues and data fragmentation. References:

? AWS Glue

? Amazon EventBridge

? Using AWS Glue to run ETL jobs against non-native JDBC data sources

? [AWS Lambda quotas]

? [Amazon Kinesis Data Firehose quotas]

NEW QUESTION 36

An airline company is collecting metrics about flight activities for analytics. The company is conducting a proof of concept (POC) test to show how analytics can provide insights that the company can use to increase on-time departures.

The POC test uses objects in Amazon S3 that contain the metrics in .csv format. The POC test uses Amazon Athena to query the data. The data is partitioned in the S3 bucket by date.

As the amount of data increases, the company wants to optimize the storage solution to improve query performance.

Which combination of solutions will meet these requirements? (Choose two.)

- A. Add a randomized string to the beginning of the keys in Amazon S3 to get more throughput across partitions.
- B. Use an S3 bucket that is in the same account that uses Athena to query the data.
- C. Use an S3 bucket that is in the same AWS Region where the company runs Athena queries.
- D. Preprocess the .csv data to JSON format by fetching only the document keys that the query requires.
- E. Preprocess the .csv data to Apache Parquet format by fetching only the data blocks that are needed for predicates.

Answer: CE

Explanation:

Using an S3 bucket that is in the same AWS Region where the company runs Athena queries can improve query performance by reducing data transfer latency

and costs. Preprocessing the .csv data to Apache Parquet format can also improve query performance by enabling columnar storage, compression, and partitioning, which can reduce the amount of data scanned and fetched by the query. These solutions can optimize the storage solution for the POC test without requiring much effort or changes to the existing data pipeline. The other solutions are not optimal or relevant for this requirement. Adding a randomized string to the beginning of the keys in Amazon S3 can improve the throughput across partitions, but it can also make the data harder to query and manage. Using an S3 bucket that is in the same account that uses Athena to query the data does not have any significant impact on query performance, as long as the proper permissions are granted. Preprocessing the .csv data to JSON format does not offer any benefits over the .csv format, as both are row-based and verbose formats that require more data scanning and fetching than columnar formats like Parquet. References:

? Best Practices When Using Athena with AWS Glue

? Optimizing Amazon S3 Performance

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 41

A company uses an Amazon Redshift cluster that runs on RA3 nodes. The company wants to scale read and write capacity to meet demand. A data engineer needs to identify a solution that will turn on concurrency scaling.

Which solution will meet this requirement?

- A. Turn on concurrency scaling in workload management (WLM) for Redshift Serverless workgroups.
- B. Turn on concurrency scaling at the workload management (WLM) queue level in the Redshift cluster.
- C. Turn on concurrency scaling in the settings during the creation of a new Redshift cluster.
- D. Turn on concurrency scaling for the daily usage quota for the Redshift cluster.

Answer: B

Explanation:

Concurrency scaling is a feature that allows you to support thousands of concurrent users and queries, with consistently fast query performance. When you turn on concurrency scaling, Amazon Redshift automatically adds query processing power in seconds to process queries without any delays. You can manage which queries are sent to the concurrency-scaling cluster by configuring WLM queues. To turn on concurrency scaling for a queue, set the Concurrency Scaling mode value to auto. The other options are either incorrect or irrelevant, as they do not enable concurrency scaling for the existing Redshift cluster on RA3 nodes.

References:

? Working with concurrency scaling - Amazon Redshift

? Amazon Redshift Concurrency Scaling - Amazon Web Services

? Configuring concurrency scaling queues - Amazon Redshift

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 6, page 163)

NEW QUESTION 42

A company stores daily records of the financial performance of investment portfolios in .csv format in an Amazon S3 bucket. A data engineer uses AWS Glue crawlers to crawl the S3 data.

The data engineer must make the S3 data accessible daily in the AWS Glue Data Catalog. Which solution will meet these requirements?

- A. Create an IAM role that includes the AmazonS3FullAccess policy
- B. Associate the role with the crawler
- C. Specify the S3 bucket path of the source data as the crawler's data store
- D. Create a daily schedule to run the crawler
- E. Configure the output destination to a new path in the existing S3 bucket.
- F. Create an IAM role that includes the AWSGlueServiceRole policy
- G. Associate the role with the crawler
- H. Specify the S3 bucket path of the source data as the crawler's data store
- I. Create a daily schedule to run the crawler
- J. Specify a database name for the output.
- K. Create an IAM role that includes the AmazonS3FullAccess policy
- L. Associate the role with the crawler
- M. Specify the S3 bucket path of the source data as the crawler's data store
- N. Allocate data processing units (DPUs) to run the crawler every day
- O. Specify a database name for the output.
- P. Create an IAM role that includes the AWSGlueServiceRole policy
- Q. Associate the role with the crawler
- R. Specify the S3 bucket path of the source data as the crawler's data store
- S. Allocate data processing units (DPUs) to run the crawler every day
- T. Configure the output destination to a new path in the existing S3 bucket.

Answer: B

Explanation:

To make the S3 data accessible daily in the AWS Glue Data Catalog, the data engineer needs to create a crawler that can crawl the S3 data and write the metadata to the Data Catalog. The crawler also needs to run on a daily schedule to keep the Data Catalog updated with the latest data. Therefore, the solution must include the following steps:

? Create an IAM role that has the necessary permissions to access the S3 data and

the Data Catalog. The AWSGlueServiceRole policy is a managed policy that grants these permissions¹.

? Associate the role with the crawler.

? Specify the S3 bucket path of the source data as the crawler's data store. The crawler will scan the data and infer the schema and format².

? Create a daily schedule to run the crawler. The crawler will run at the specified time every day and update the Data Catalog with any changes in the data³.

? Specify a database name for the output. The crawler will create or update a table in the Data Catalog under the specified database. The table will contain the metadata about the data in the S3 bucket, such as the location, schema, and classification.

Option B is the only solution that includes all these steps. Therefore, option B is the correct answer.

Option A is incorrect because it configures the output destination to a new path in the existing S3 bucket. This is unnecessary and may cause confusion, as the crawler does not write any data to the S3 bucket, only metadata to the Data Catalog.

Option C is incorrect because it allocates data processing units (DPUs) to run the crawler every day. This is also unnecessary, as DPUs are only used for AWS Glue ETL jobs, not crawlers.

Option D is incorrect because it combines the errors of option A and C. It configures the output destination to a new path in the existing S3 bucket and allocates DPUs to run the crawler every day, both of which are irrelevant for the crawler.

References:

- ? 1: AWS managed (predefined) policies for AWS Glue - AWS Glue
- ? 2: Data Catalog and crawlers in AWS Glue - AWS Glue
- ? 3: Scheduling an AWS Glue crawler - AWS Glue
- ? [4]: Parameters set on Data Catalog tables by crawler - AWS Glue
- ? [5]: AWS Glue pricing - Amazon Web Services (AWS)

NEW QUESTION 44

A financial company wants to use Amazon Athena to run on-demand SQL queries on a petabyte-scale dataset to support a business intelligence (BI) application. An AWS Glue job that runs during non-business hours updates the dataset once every day. The BI application has a standard data refresh frequency of 1 hour to comply with company policies.

A data engineer wants to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs.

Which solution will meet these requirements with the LEAST operational overhead?

- A. Configure an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day
- B. Use the query result reuse feature of Amazon Athena for the SQL queries.
- C. Add an Amazon ElastiCache cluster between the BI application and Athena.
- D. Change the format of the files that are in the dataset to Apache Parquet.

Answer: B

Explanation:

The best solution to cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs is to use the query result reuse feature of Amazon Athena for the SQL queries. This feature allows you to run the same query multiple times without incurring additional charges, as long as the underlying data has not changed and the query results are still in the query result location in Amazon S3. This feature is useful for scenarios where you have a petabyte-scale dataset that is updated infrequently, such as once a day, and you have a BI application that runs the same queries repeatedly, such as every hour. By using the query result reuse feature, you can reduce the amount of data scanned by your queries and save on the cost of running Athena. You can enable or disable this feature at the workgroup level or at the individual query level¹.

Option A is not the best solution, as configuring an Amazon S3 Lifecycle policy to move data to the S3 Glacier Deep Archive storage class after 1 day would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon S3 Lifecycle policies are rules that you can define to automatically transition objects between different storage classes based on specified criteria, such as the age of the object². S3 Glacier Deep Archive is the lowest-cost storage class in Amazon S3, designed for long-term data archiving that is accessed once or twice in a year³. While moving data to S3 Glacier Deep Archive can reduce the storage cost, it would also increase the retrieval cost and latency, as it takes up to 12 hours to restore the data from S3 Glacier Deep Archive³. Moreover, Athena does not support querying data that is in S3 Glacier or S3 Glacier Deep Archive storage classes⁴. Therefore, using this option would not meet the requirements of running on-demand SQL queries on the dataset.

Option C is not the best solution, as adding an Amazon ElastiCache cluster between the BI application and Athena would not cost optimize the company's use of Amazon Athena, but rather increase the cost and complexity. Amazon ElastiCache is a service that offers fully managed in-memory data stores, such as Redis and Memcached, that can improve the performance and scalability of web applications by caching frequently accessed data. While using ElastiCache can reduce the latency and load on the BI application, it would not reduce the amount of data scanned by Athena, which is the main factor that determines the cost of running Athena. Moreover, using ElastiCache would introduce additional infrastructure costs and operational overhead, as you would have to provision, manage, and scale the ElastiCache cluster, and integrate it with the BI application and Athena. Option D is not the best solution, as changing the format of the files that are in the dataset to Apache Parquet would not cost optimize the company's use of Amazon Athena without adding any additional infrastructure costs, but rather increase the complexity. Apache Parquet is a columnar storage format that can improve the performance of analytical queries by reducing the amount of data that needs to be scanned and providing efficient compression and encoding schemes. However, changing the format of the files that are in the dataset to Apache Parquet would require additional processing and transformation steps, such as using AWS Glue or Amazon EMR to convert the files from their original format to Parquet, and storing the converted files in a separate location in Amazon S3. This would increase the complexity and the operational overhead of the data pipeline, and also incur additional costs for using AWS Glue or Amazon EMR. References:

- ? Query result reuse
- ? Amazon S3 Lifecycle
- ? S3 Glacier Deep Archive
- ? Storage classes supported by Athena
- ? [What is Amazon ElastiCache?]
- ? [Amazon Athena pricing]
- ? [Columnar Storage Formats]
- ? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide

NEW QUESTION 49

A data engineer must ingest a source of structured data that is in .csv format into an Amazon S3 data lake. The .csv files contain 15 columns. Data analysts need to run Amazon Athena queries on one or two columns of the dataset. The data analysts rarely query the entire file.

Which solution will meet these requirements MOST cost-effectively?

- A. Use an AWS Glue PySpark job to ingest the source data into the data lake in .csv format.
- B. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source.
- C. Configure the job to ingest the data into the data lake in JSON format.
- D. Use an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format.
- E. Create an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source.
- F. Configure the job to write the data into the data lake in Apache Parquet format.

Answer: D

Explanation:

Amazon Athena is a serverless interactive query service that allows you to analyze data in Amazon S3 using standard SQL. Athena supports various data formats, such as CSV, JSON, ORC, Avro, and Parquet. However, not all data formats are equally efficient for querying. Some data formats, such as CSV and JSON, are row-oriented, meaning that they store data as a sequence of records, each with the same fields. Row-oriented formats are suitable for loading and exporting data, but they are not optimal for analytical queries that often access only a subset of columns. Row-oriented formats also do not support compression or encoding techniques that can reduce the data size and improve the query performance.

On the other hand, some data formats, such as ORC and Parquet, are column-oriented, meaning that they store data as a collection of columns, each with a specific data type. Column-oriented formats are ideal for analytical queries that often filter, aggregate, or join data by columns. Column-oriented formats also support compression and encoding techniques that can reduce the data size and improve the query performance. For example, Parquet supports dictionary encoding, which replaces repeated values with numeric codes, and run-length encoding, which replaces consecutive identical values with a single value and a count. Parquet also supports various compression algorithms, such as Snappy, GZIP, and ZSTD, that can further reduce the data size and improve the query

performance.

Therefore, creating an AWS Glue extract, transform, and load (ETL) job to read from the .csv structured data source and writing the data into the data lake in Apache Parquet format will meet the requirements most cost-effectively. AWS Glue is a fully managed service that provides a serverless data integration platform for data preparation, data cataloging, and data loading. AWS Glue ETL jobs allow you to transform and load data from various sources into various targets, using either a graphical interface (AWS Glue Studio) or a code-based interface (AWS Glue console or AWS Glue API). By using AWS Glue ETL jobs, you can easily convert the data from CSV to Parquet format, without having to write or manage any code. Parquet is a column-oriented format that allows Athena to scan only the relevant columns and skip the rest, reducing the amount of data read from S3. This solution will also reduce the cost of Athena queries, as Athena charges based on the amount of data scanned from S3.

The other options are not as cost-effective as creating an AWS Glue ETL job to write the data into the data lake in Parquet format. Using an AWS Glue PySpark job to ingest the source data into the data lake in .csv format will not improve the query performance or reduce the query cost, as .csv is a row-oriented format that does not support columnar access or compression. Creating an AWS Glue ETL job to ingest the data into the data lake in JSON format will not improve the query performance or reduce the query cost, as JSON is also a row-oriented format that does not support columnar access or compression. Using an AWS Glue PySpark job to ingest the source data into the data lake in Apache Avro format will improve the query performance, as Avro is a column-oriented format that supports compression and encoding, but it will require more operational effort, as you will need to write and maintain PySpark code to convert the data from CSV to Avro format. References:

? Amazon Athena

? Choosing the Right Data Format

? AWS Glue

? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide], Chapter 5: Data Analysis and Visualization, Section 5.1: Amazon Athena

NEW QUESTION 54

A company is developing an application that runs on Amazon EC2 instances. Currently, the data that the application generates is temporary. However, the company needs to persist the data, even if the EC2 instances are terminated.

A data engineer must launch new EC2 instances from an Amazon Machine Image (AMI) and configure the instances to preserve the data.

Which solution will meet this requirement?

- A. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data
- B. Apply the default settings to the EC2 instances.
- C. Launch new EC2 instances by using an AMI that is backed by a root Amazon Elastic Block Store (Amazon EBS) volume that contains the application data
- D. Apply the default settings to the EC2 instances.
- E. Launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume
- F. Attach an Amazon Elastic Block Store (Amazon EBS) volume to contain the application data
- G. Apply the default settings to the EC2 instances.
- H. Launch new EC2 instances by using an AMI that is backed by an Amazon Elastic Block Store (Amazon EBS) volume
- I. Attach an additional EC2 instance store volume to contain the application data
- J. Apply the default settings to the EC2 instances.

Answer: C

Explanation:

Amazon EC2 instances can use two types of storage volumes: instance store volumes and Amazon EBS volumes. Instance store volumes are ephemeral, meaning they are only attached to the instance for the duration of its life cycle. If the instance is stopped, terminated, or fails, the data on the instance store volume is lost. Amazon EBS volumes are persistent, meaning they can be detached from the instance and attached to another instance, and the data on the volume is preserved. To meet the requirement of persisting the data even if the EC2 instances are terminated, the data engineer must use Amazon EBS volumes to store the application data. The solution is to launch new EC2 instances by using an AMI that is backed by an EC2 instance store volume, which is the default option for most AMIs. Then, the data engineer must attach an Amazon EBS volume to each instance and configure the application to write the data to the EBS volume. This way, the data will be saved on the EBS volume and can be accessed by another instance if needed. The data engineer can apply the default settings to the EC2 instances, as there is no need to modify the instance type, security group, or IAM role for this solution. The other options are either not feasible or not optimal. Launching new EC2 instances by using an AMI that is backed by an EC2 instance store volume that contains the application data (option A) or by using an AMI that is backed by a root Amazon EBS volume that contains the application data (option B) would not work, as the data on the AMI would be outdated and overwritten by the new instances. Attaching an additional EC2 instance store volume to contain the application data (option D) would not work, as the data on the instance store volume would be lost if the instance is terminated. References:

? Amazon EC2 Instance Store

? Amazon EBS Volumes

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 2: Data Store Management, Section 2.1: Amazon EC2

NEW QUESTION 58

A company uses an Amazon Redshift provisioned cluster as its database. The Redshift cluster has five reserved ra3.4xlarge nodes and uses key distribution.

A data engineer notices that one of the nodes frequently has a CPU load over 90%. SQL Queries that run on the node are queued. The other four nodes usually have a CPU load under 15% during daily operations.

The data engineer wants to maintain the current number of compute nodes. The data engineer also wants to balance the load more evenly across all five compute nodes.

Which solution will meet these requirements?

- A. Change the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.
- B. Change the distribution key to the table column that has the largest dimension.
- C. Upgrade the reserved node from ra3.4xlarge to ra3.16xlarge.
- D. Change the primary key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement.

Answer: B

Explanation:

Changing the distribution key to the table column that has the largest dimension will help to balance the load more evenly across all five compute nodes. The distribution key determines how the rows of a table are distributed among the slices of the cluster. If the distribution key is not chosen wisely, it can cause data skew, meaning some slices will have more data than others, resulting in uneven CPU load and query performance. By choosing the table column that has the largest dimension, meaning the column that has the most distinct values, as the distribution key, the data engineer can ensure that the rows are distributed more uniformly across the slices, reducing data skew and improving query performance.

The other options are not solutions that will meet the requirements. Option A, changing the sort key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load. The sort key determines the order in which the rows of a table are stored on disk, which can improve the performance of range-restricted queries, but not the load balancing. Option C, upgrading the reserved node from ra3.4xlarge to ra3.16xlarge, will not maintain the current number of compute nodes, as it will increase the cost and the capacity of the cluster. Option D, changing the primary

key to be the data column that is most often used in a WHERE clause of the SQL SELECT statement, will not affect the data distribution or the CPU load either. The primary key is a constraint that enforces the uniqueness of the rows in a table, but it does not influence the data layout or the query optimization. References:

- ? Choosing a data distribution style
- ? Choosing a data sort key
- ? Working with primary keys

NEW QUESTION 62

A data engineer needs to securely transfer 5 TB of data from an on-premises data center to an Amazon S3 bucket. Approximately 5% of the data changes every day. Updates to the data need to be regularly proliferated to the S3 bucket. The data includes files that are in multiple formats. The data engineer needs to automate the transfer process and must schedule the process to run periodically.

Which AWS service should the data engineer use to transfer the data in the MOST operationally efficient way?

- A. AWS DataSync
- B. AWS Glue
- C. AWS Direct Connect
- D. Amazon S3 Transfer Acceleration

Answer: A

Explanation:

AWS DataSync is an online data movement and discovery service that simplifies and accelerates data migrations to AWS as well as moving data to and from on-premises storage, edge locations, other cloud providers, and AWS Storage services¹. AWS DataSync can copy data to and from various sources and targets, including Amazon S3, and handle files in multiple formats. AWS DataSync also supports incremental transfers, meaning it can detect and copy only the changes to the data, reducing the amount of data transferred and improving the performance. AWS DataSync can automate and schedule the transfer process using triggers, and monitor the progress and status of the transfers using CloudWatch metrics and events¹.

AWS DataSync is the most operationally efficient way to transfer the data in this scenario, as it meets all the requirements and offers a serverless and scalable solution. AWS Glue, AWS Direct Connect, and Amazon S3 Transfer Acceleration are not the best options for this scenario, as they have some limitations or drawbacks compared to AWS DataSync. AWS Glue is a serverless ETL service that can extract, transform, and load data from various sources to various targets, including Amazon S3². However, AWS Glue is not designed for large-scale data transfers, as it has some quotas and limits on the number and size of files it can process³. AWS Glue also does not support incremental transfers, meaning it would have to copy the entire data set every time, which would be inefficient and costly.

AWS Direct Connect is a service that establishes a dedicated network connection between your on-premises data center and AWS, bypassing the public internet and improving the bandwidth and performance of the data transfer. However, AWS Direct Connect is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS DataSync, AWS Storage Gateway, or AWS CLI. AWS Direct Connect also has some hardware and location requirements, and charges you for the port hours and data transfer out of AWS.

Amazon S3 Transfer Acceleration is a feature that enables faster data transfers to Amazon S3 over long distances, using the AWS edge locations and optimized network paths. However, Amazon S3 Transfer Acceleration is not a data transfer service by itself, as it requires additional services or tools to copy the data, such as AWS CLI, AWS SDK, or third-party software. Amazon S3 Transfer Acceleration also charges you for the data transferred over the accelerated endpoints, and does not guarantee a performance improvement for every transfer, as it depends on various factors such as the network conditions, the distance, and the object size. References:

- ? AWS DataSync
- ? AWS Glue
- ? AWS Glue quotas and limits
- ? [AWS Direct Connect]
- ? [Data transfer options for AWS Direct Connect]
- ? [Amazon S3 Transfer Acceleration]
- ? [Using Amazon S3 Transfer Acceleration]

NEW QUESTION 65

A data engineer is configuring an AWS Glue job to read data from an Amazon S3 bucket. The data engineer has set up the necessary AWS Glue connection details and an associated IAM role. However, when the data engineer attempts to run the AWS Glue job, the data engineer receives an error message that indicates that there are problems with the Amazon S3 VPC gateway endpoint.

The data engineer must resolve the error and connect the AWS Glue job to the S3 bucket. Which solution will meet this requirement?

- A. Update the AWS Glue security group to allow inbound traffic from the Amazon S3 VPC gateway endpoint.
- B. Configure an S3 bucket policy to explicitly grant the AWS Glue job permissions to access the S3 bucket.
- C. Review the AWS Glue job code to ensure that the AWS Glue connection details include a fully qualified domain name.
- D. Verify that the VPC's route table includes inbound and outbound routes for the Amazon S3 VPC gateway endpoint.

Answer: D

Explanation:

The error message indicates that the AWS Glue job cannot access the Amazon S3 bucket through the VPC endpoint. This could be because the VPC's route table does not have the necessary routes to direct the traffic to the endpoint. To fix this, the data engineer must verify that the route table has an entry for the Amazon S3 service prefix (com.amazonaws.region.s3) with the target as the VPC endpoint ID. This will allow the AWS Glue job to use the VPC endpoint to access the S3 bucket without going through the internet or a NAT gateway. For more information, see Gateway endpoints^R. References:

- ? Troubleshoot the AWS Glue error "VPC S3 endpoint validation failed"
- ? Amazon VPC endpoints for Amazon S3
- ? [AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide]

NEW QUESTION 67

A company is planning to use a provisioned Amazon EMR cluster that runs Apache Spark jobs to perform big data analysis. The company requires high reliability. A big data team must follow best practices for running cost-optimized and long-running workloads on Amazon EMR. The team must find a solution that will maintain the company's current level of performance.

Which combination of resources will meet these requirements MOST cost-effectively? (Choose two.)

- A. Use Hadoop Distributed File System (HDFS) as a persistent data store.
- B. Use Amazon S3 as a persistent data store.
- C. Use x86-based instances for core nodes and task nodes.
- D. Use Graviton instances for core nodes and task nodes.

E. Use Spot Instances for all primary nodes.

Answer: BD

Explanation:

The best combination of resources to meet the requirements of high reliability, cost-optimization, and performance for running Apache Spark jobs on Amazon EMR is to use Amazon S3 as a persistent data store and Graviton instances for core nodes and task nodes.

Amazon S3 is a highly durable, scalable, and secure object storage service that can store any amount of data for a variety of use cases, including big data analytics¹. Amazon S3 is a better choice than HDFS as a persistent data store for Amazon EMR, as it decouples the storage from the compute layer, allowing for more flexibility and cost-efficiency. Amazon S3 also supports data encryption, versioning, lifecycle management, and cross-region replication¹. Amazon EMR integrates seamlessly with Amazon S3, using EMR File System (EMRFS) to access data stored in Amazon S3 buckets². EMRFS also supports consistent view, which enables Amazon EMR to provide read-after-write consistency for Amazon S3 objects that are accessed through EMRFS².

Graviton instances are powered by Arm-based AWS Graviton² processors that deliver up to 40% better price performance over comparable current generation x86-based instances³. Graviton instances are ideal for running workloads that are CPU-bound, memory-bound, or network-bound, such as big data analytics, web servers, and open- source databases³. Graviton instances are compatible with Amazon EMR, and can be used for both core nodes and task nodes. Core nodes are responsible for running the data processing frameworks, such as Apache Spark, and storing data in HDFS or the local file system. Task nodes are optional nodes that can be added to a cluster to increase the processing power and throughput. By using Graviton instances for both core nodes and task nodes, you can achieve higher performance and lower cost than using x86-based instances.

Using Spot Instances for all primary nodes is not a good option, as it can compromise the reliability and availability of the cluster. Spot Instances are spare EC2 instances that are available at up to 90% discount compared to On-Demand prices, but they can be interrupted by EC2 with a two-minute notice when EC2 needs the capacity back. Primary nodes are the nodes that run the cluster software, such as Hadoop, Spark, Hive, and Hue, and are essential for the cluster operation. If a primary node is interrupted by EC2, the cluster will fail or become unstable. Therefore, it is recommended to use On-Demand Instances or Reserved Instances for primary nodes, and use Spot Instances only for task nodes that can tolerate interruptions. References:

? Amazon S3 - Cloud Object Storage

? EMR File System (EMRFS)

? AWS Graviton² Processor-Powered Amazon EC2 Instances

? [Plan and Configure EC2 Instances]

? [Amazon EC2 Spot Instances]

? [Best Practices for Amazon EMR]

NEW QUESTION 69

A data engineer needs to maintain a central metadata repository that users access through Amazon EMR and Amazon Athena queries. The repository needs to provide the schema and properties of many tables. Some of the metadata is stored in Apache Hive. The data engineer needs to import the metadata from Hive into the central metadata repository.

Which solution will meet these requirements with the LEAST development effort?

A. Use Amazon EMR and Apache Ranger.

B. Use a Hive metastore on an EMR cluster.

C. Use the AWS Glue Data Catalog.

D. Use a metastore on an Amazon RDS for MySQL DB instance.

Answer: C

Explanation:

The AWS Glue Data Catalog is an Apache Hive metastore-compatible catalog that provides a central metadata repository for various data sources and formats. You can use the AWS Glue Data Catalog as an external Hive metastore for Amazon EMR and Amazon Athena queries, and import metadata from existing Hive metastores into the Data Catalog. This solution requires the least development effort, as you can use AWS Glue crawlers to automatically discover and catalog the metadata from Hive, and use the AWS Glue console, AWS CLI, or Amazon EMR API to configure the Data Catalog as the Hive metastore. The other options are either more complex or require additional steps, such as setting up Apache Ranger for security, managing a Hive metastore on an EMR cluster or an RDS instance, or migrating the metadata manually. References:

? Using the AWS Glue Data Catalog as the metastore for Hive (Section: Specifying

AWS Glue Data Catalog as the metastore)

? Metadata Management: Hive Metastore vs AWS Glue (Section: AWS Glue Data Catalog)

? AWS Glue Data Catalog support for Spark SQL jobs (Section: Importing metadata from an existing Hive metastore)

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide (Chapter 5, page 131)

NEW QUESTION 72

A data engineer needs to use AWS Step Functions to design an orchestration workflow. The workflow must parallel process a large collection of data files and apply a specific transformation to each file.

Which Step Functions state should the data engineer use to meet these requirements?

A. Parallel state

B. Choice state

C. Map state

D. Wait state

Answer: C

Explanation:

Option C is the correct answer because the Map state is designed to process a collection of data in parallel by applying the same transformation to each element. The Map state can invoke a nested workflow for each element, which can be another state machine or a Lambda function. The Map state will wait until all the parallel executions are completed before moving to the next state.

Option A is incorrect because the Parallel state is used to execute multiple branches of logic concurrently, not to process a collection of data. The Parallel state can have different branches with different logic and states, whereas the Map state has only one branch that is applied to each element of the collection.

Option B is incorrect because the Choice state is used to make decisions based on a comparison of a value to a set of rules. The Choice state does not process any data or invoke any nested workflows.

Option D is incorrect because the Wait state is used to delay the state machine from continuing for a specified time. The Wait state does not process any data or invoke any nested workflows.

References:

? AWS Certified Data Engineer - Associate DEA-C01 Complete Study Guide, Chapter 5: Data Orchestration, Section 5.3: AWS Step Functions, Pages 131-132

? Building Batch Data Analytics Solutions on AWS, Module 5: Data Orchestration, Lesson 5.2: AWS Step Functions, Pages 9-10

? AWS Documentation Overview, AWS Step Functions Developer Guide, Step Functions Concepts, State Types, Map State, Pages 1-3

NEW QUESTION 76

.....

Thank You for Trying Our Product

We offer two products:

1st - We have Practice Tests Software with Actual Exam Questions

2nd - Questions and Answers in PDF Format

AWS-Certified-Data-Engineer-Associate Practice Exam Features:

- * AWS-Certified-Data-Engineer-Associate Questions and Answers Updated Frequently
- * AWS-Certified-Data-Engineer-Associate Practice Questions Verified by Expert Senior Certified Staff
- * AWS-Certified-Data-Engineer-Associate Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- * AWS-Certified-Data-Engineer-Associate Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year

100% Actual & Verified — Instant Download, Please Click
[Order The AWS-Certified-Data-Engineer-Associate Practice Test Here](#)